Developing Executable Specifications for Networking Smart Transducers to Bluetooth

Steven B. Bibyk

Information Electronics

The Ohio State University

bibyk.1@osu.edu

Problem Statement

- Networking smart transducers wirelessly is attractive
- For small-span networks, using existing wireless technology is simpler, quicker and inexpensive
- Challenge is to design a *network infrastructure* according to voluminous, complicated standards and new commercial technology

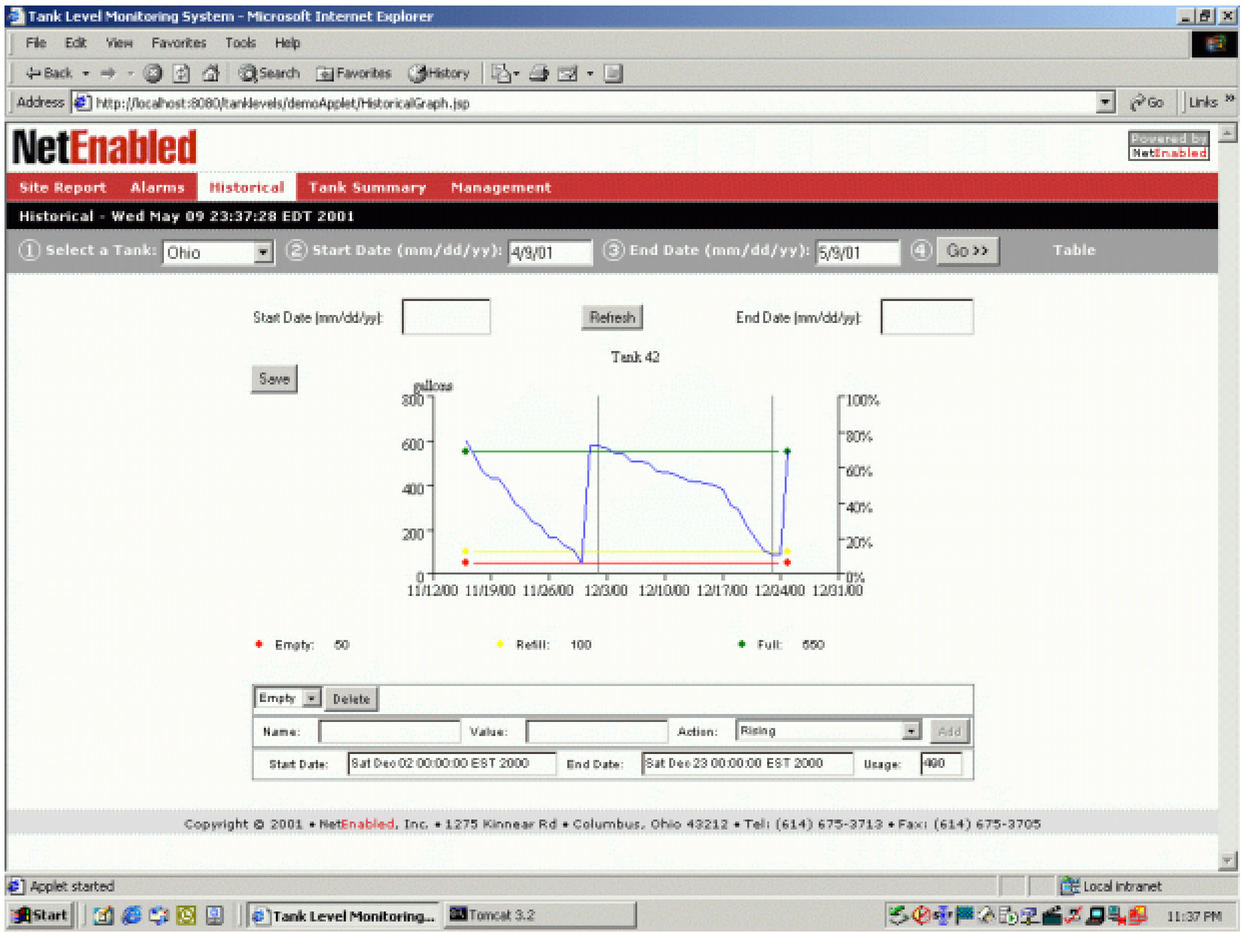
Wireless Sensor Standards

- Combining Sensor Standards with Wireless Standards is error prone. Key requirements are nebulous --- battery life?
- Need to be able to exercise & forecast how the specifications will play out:
  -System Level Definition Language
  www.sldl.org (get white paper)
- Hardware Description Language (VHDL)

NetEnabled Overview

- Connecting Devices to:
  - Networks
  - Data centers
  - Desktops
- Using:
  - Serial, Ethernet, RF, TCP/IP, cellular, CDPD, satellite, pager
- Need:
  - RF technologies supporting:
    - 4+ years for hourly readings using only 3.6V, 5000mAhr battery
    - Variable transmissions power and distance
    - Fast sync times and short protocol negotiations
    - Application: Class I, Div I environments

MyTankLevels.com



Department of Electrical Engineering

HOME

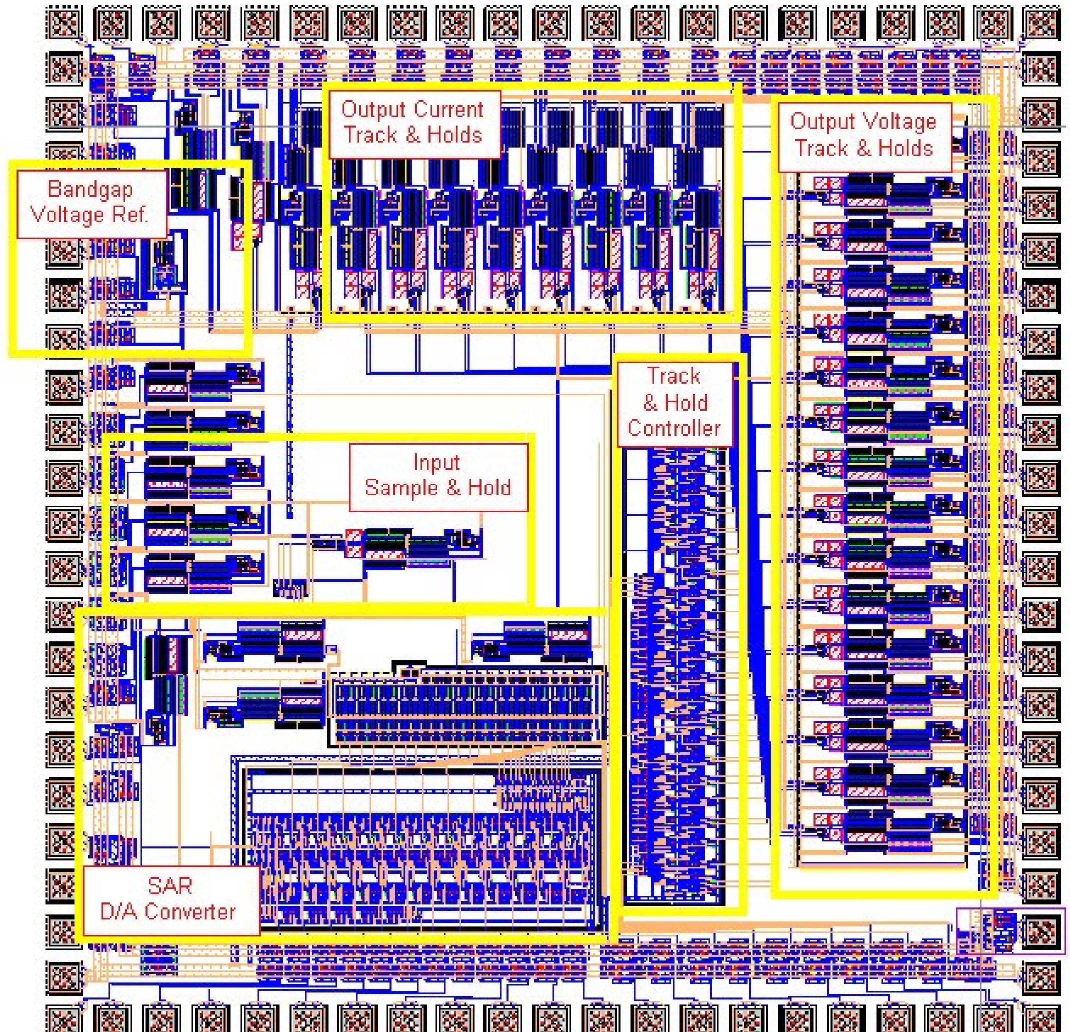
ASIC+

- Electronics Design Business
- Specialty in Mixed Signal ASIC tapeouts.
- RF CMOS chip design and delivery.
- Embedded Systems and Firmware.
- Strong Methodology in using Hardware Description Languages - mainly for digital systems but also for Analog
- IEEE 1076 (1987, ‘93, ‘99) VHDL-AMS

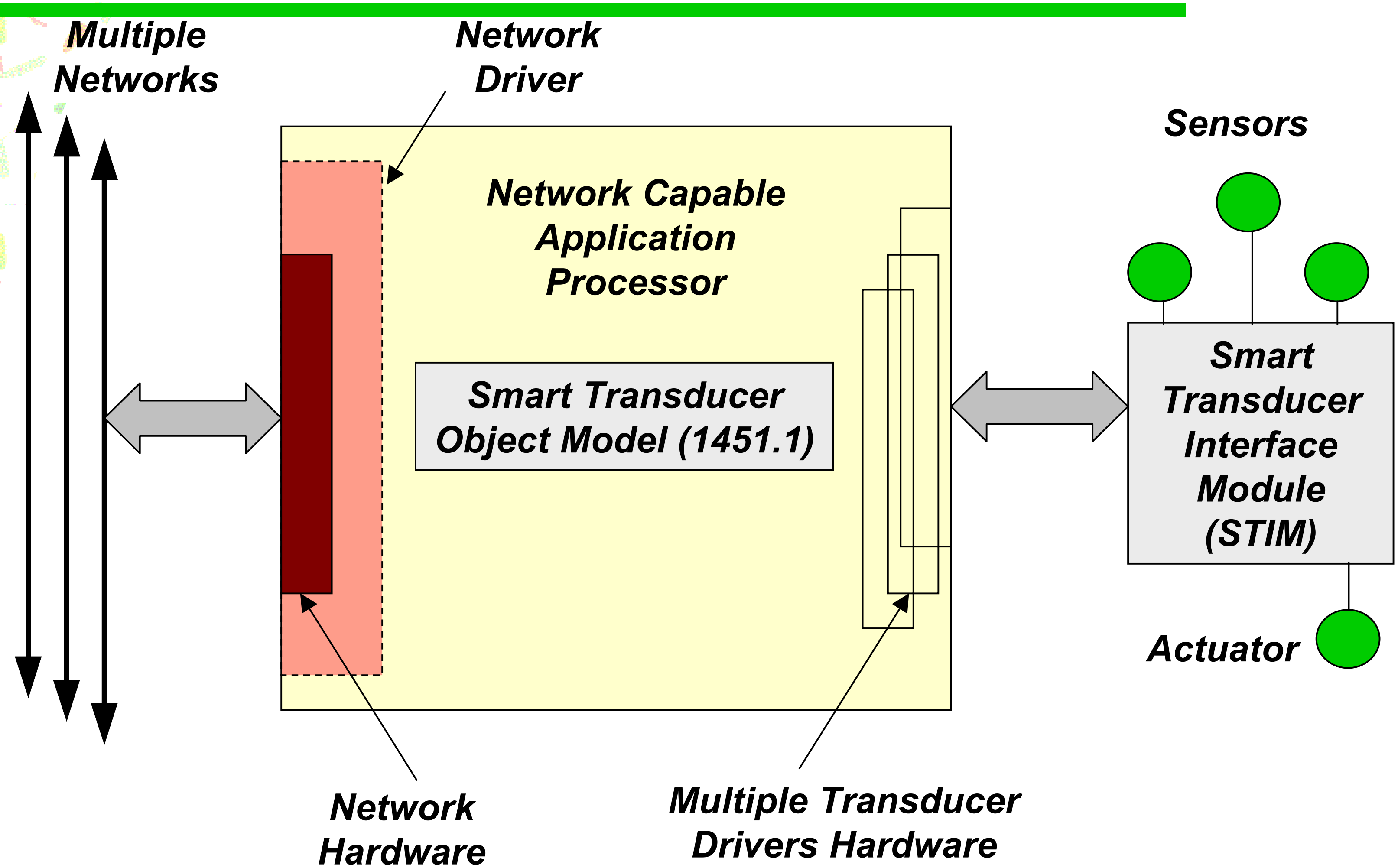
Past Designs and Customers

- System (SoC)
- RF Sensor Codec
- Satellite Control
- RF ID Tag
- UWB

- contact:
- Todd James



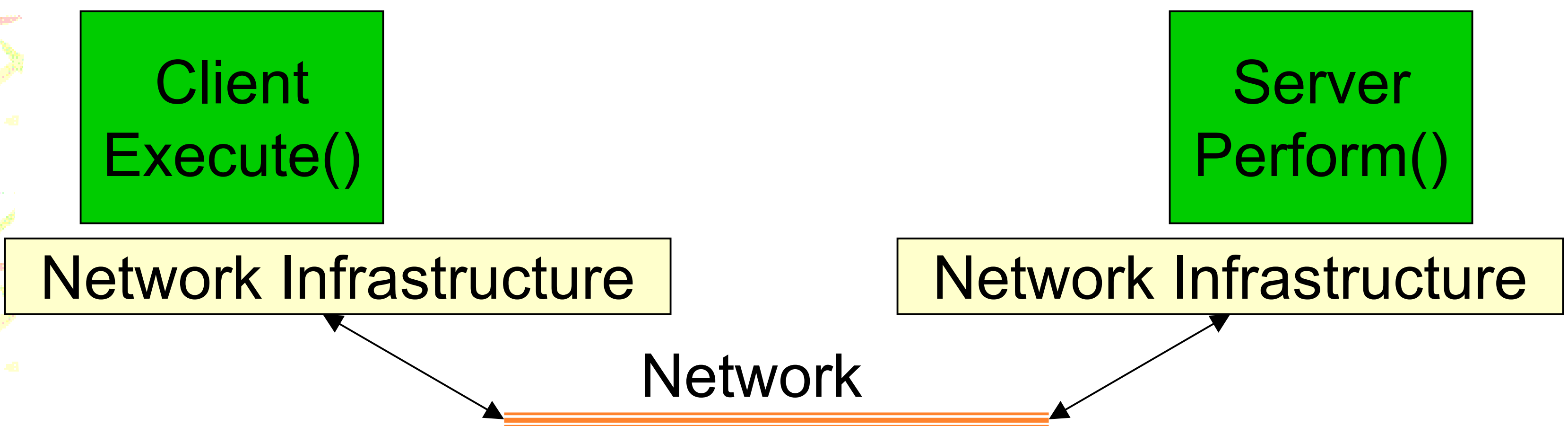
IEEE 1451.1 Standard



Background

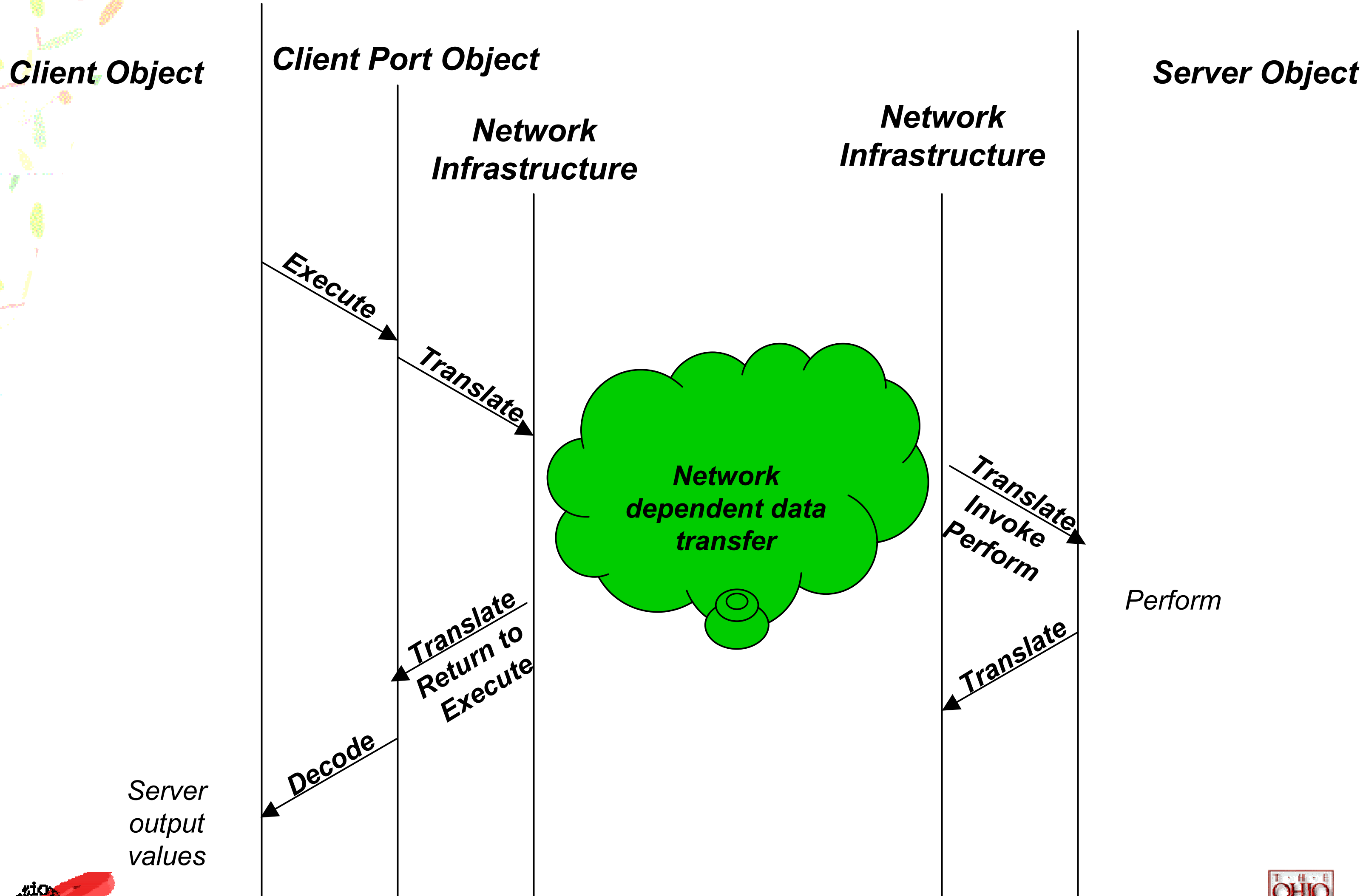
IEEE 1451.1 Client-Server Network Model

- Tightly coupled, point-to-point model
- NetInf translates IEEE 1451 datums to network specific format

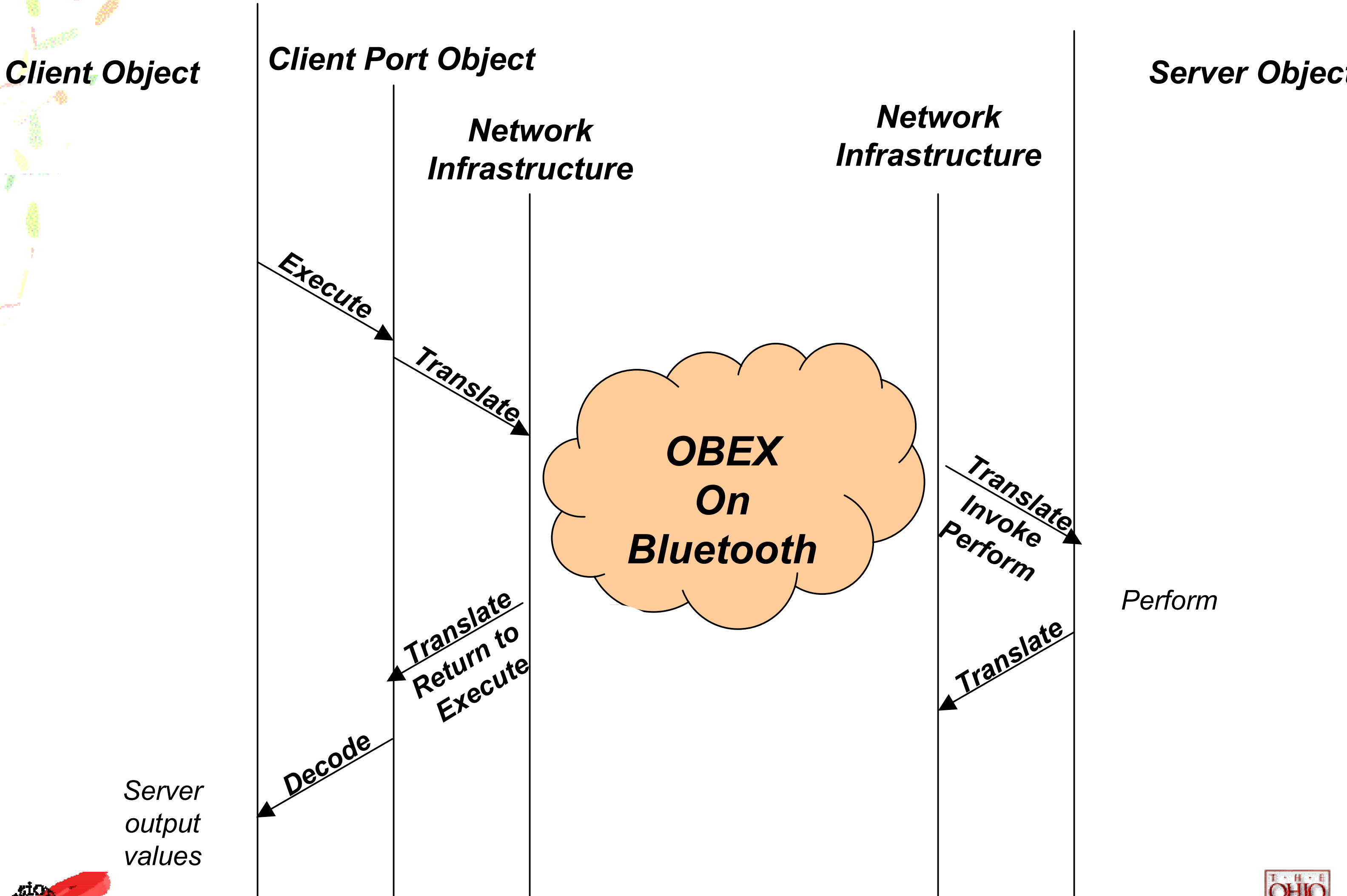


HOME

IEEE 1451 Client-Server Model



IEEE 1451 Communication on Bluetooth

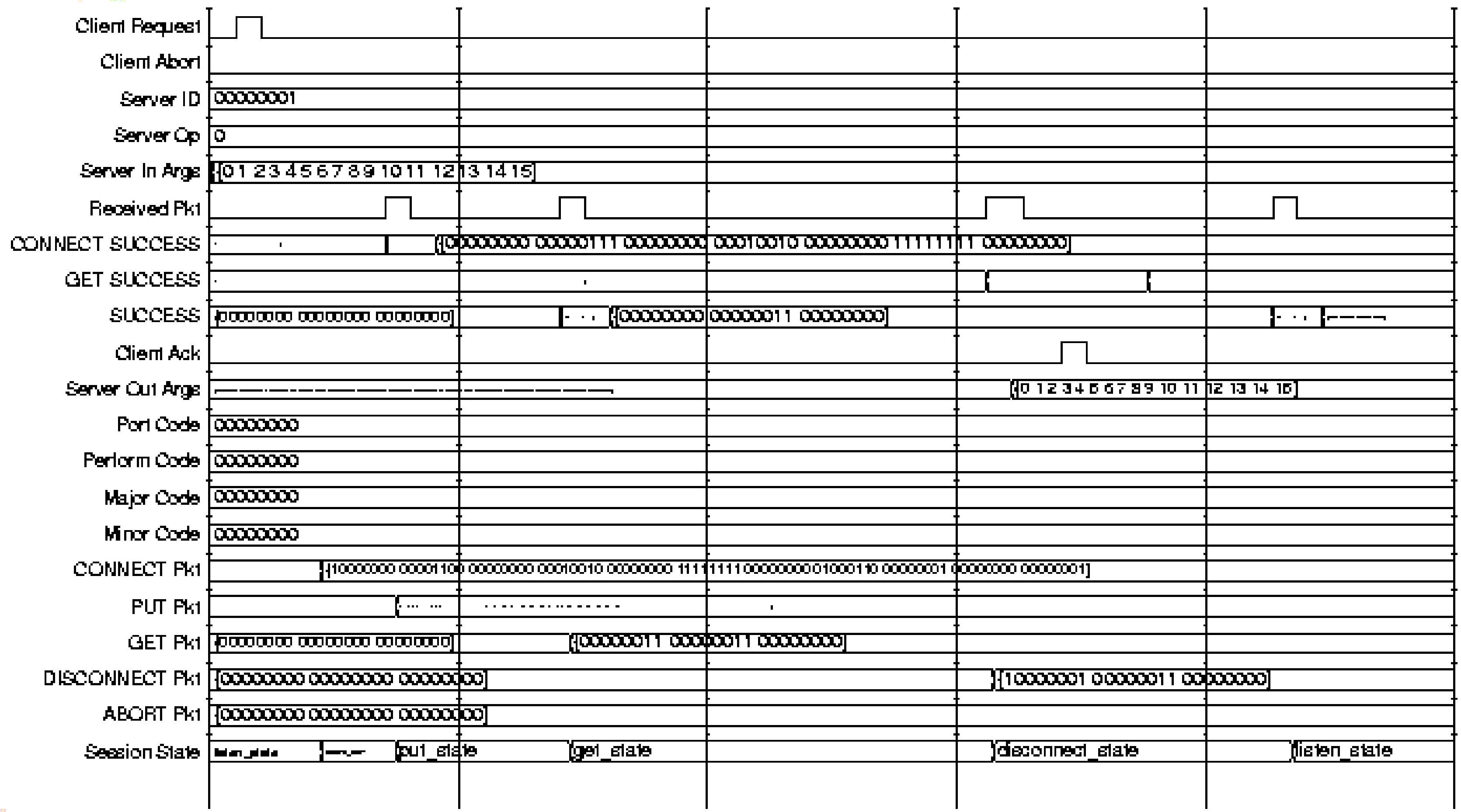


VHDL Design of Interface Entity

Problem Statement

```
ENTITY interface IS
PORT(
   client_request : IN BIT;
   client_abort : IN BIT;
   serverid : IN UInteger8;
   server_operation : IN INTEGER;
   server_input_args : IN integer_array;
   received_pkt : IN BIT;
   connect_success_pkt : IN byte_array(0 TO 6);
   get_success_pkt : IN byte_array(0 TO payload_packet_length);
   success_pkt : IN byte_array(0 TO 2);
   client_ack : OUT BIT;
   server_output_args : OUT integer_array;
   portcode : OUT UInteger8;
   performcode : OUT UInteger8;
   majorcode : OUT UInteger8;
   minorcode : OUT UInteger8;
   connect_pkt : OUT byte_array(0 TO 10);
   put_pkt : OUT byte_array(0 TO payload_packet_length);
   get_pkt : OUT byte_array(0 TO 2);
   disconnect_pkt : OUT byte_array(0 TO 2);
   abort_pkt : OUT byte_array(0 TO 2);
   session_state : OUT state);
END interface;
```

Example VHDL Simulation



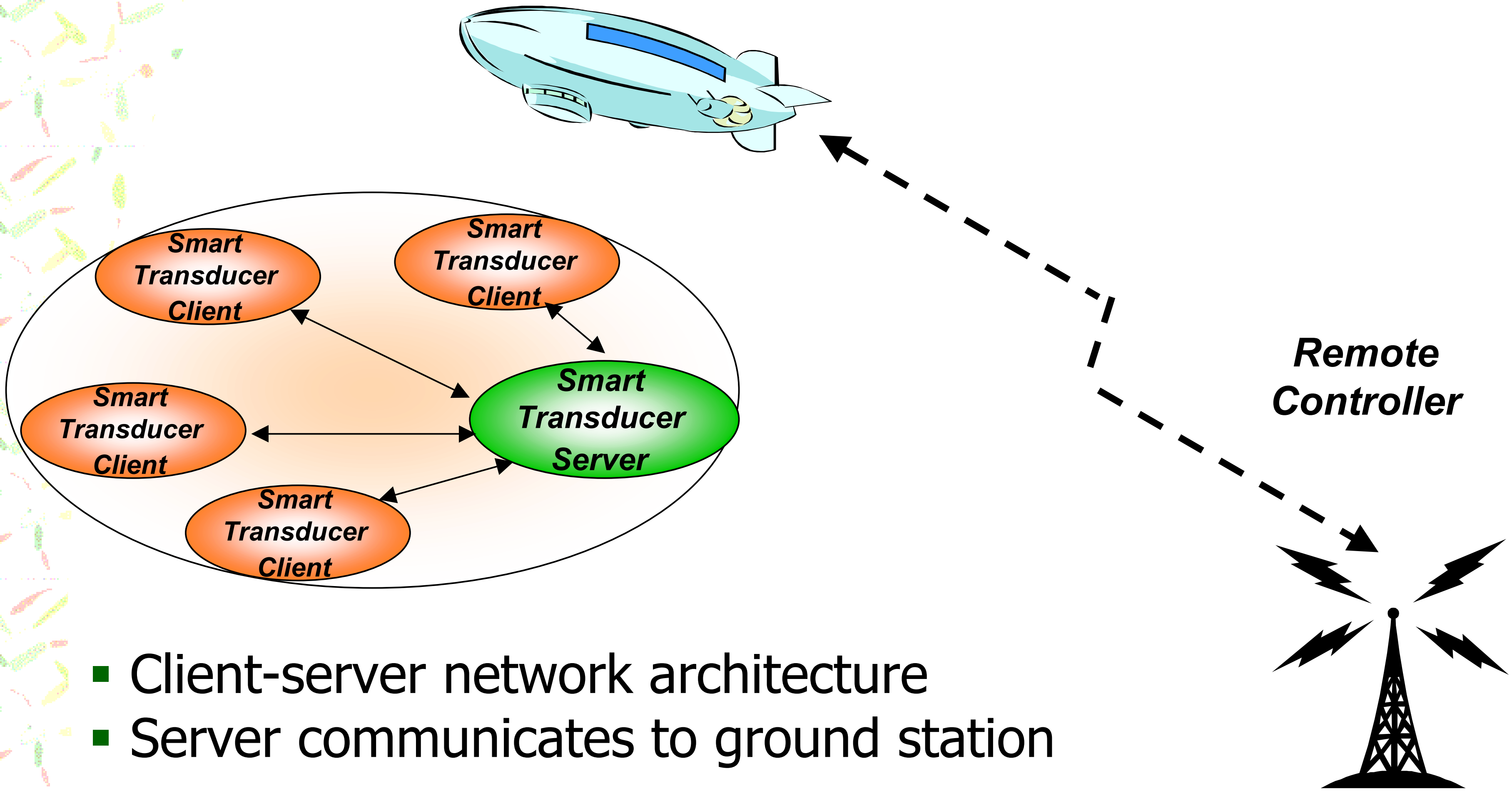
Methodology

- Design the requirements of a network infrastructure.
  - Detailed study of IEEE 1451 standard
  - Evaluation of existing wireless technology examination of Bluetooth
  - Detailed study of a session protocol, OBEX
  - How should IEEE 1451 transducers communicate on OBEX?
- Develop a VHDL software model of system.
  - Behavioral description of client-side functionality.
  - Simulations

Example of a Smart Transducer Network

- Remotely monitored UAV (Unmanned Aerial Vehicle)
- Applications
  - Military surveillance and law enforcement
  - Environment monitoring and pollution control
  - Media coverage

Example of a Smart Transducer Network

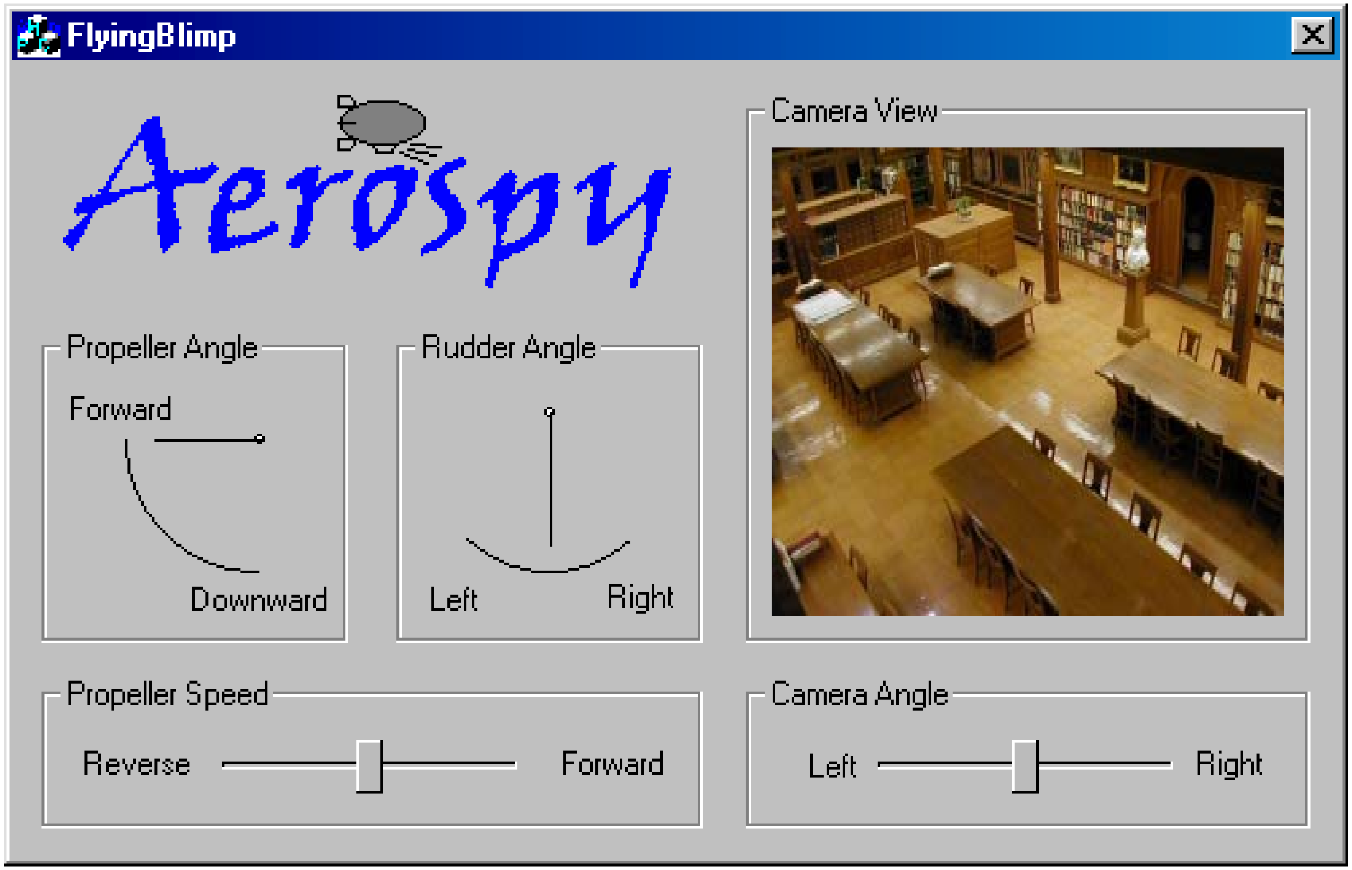


- Client-server network architecture
- Server communicates to ground station

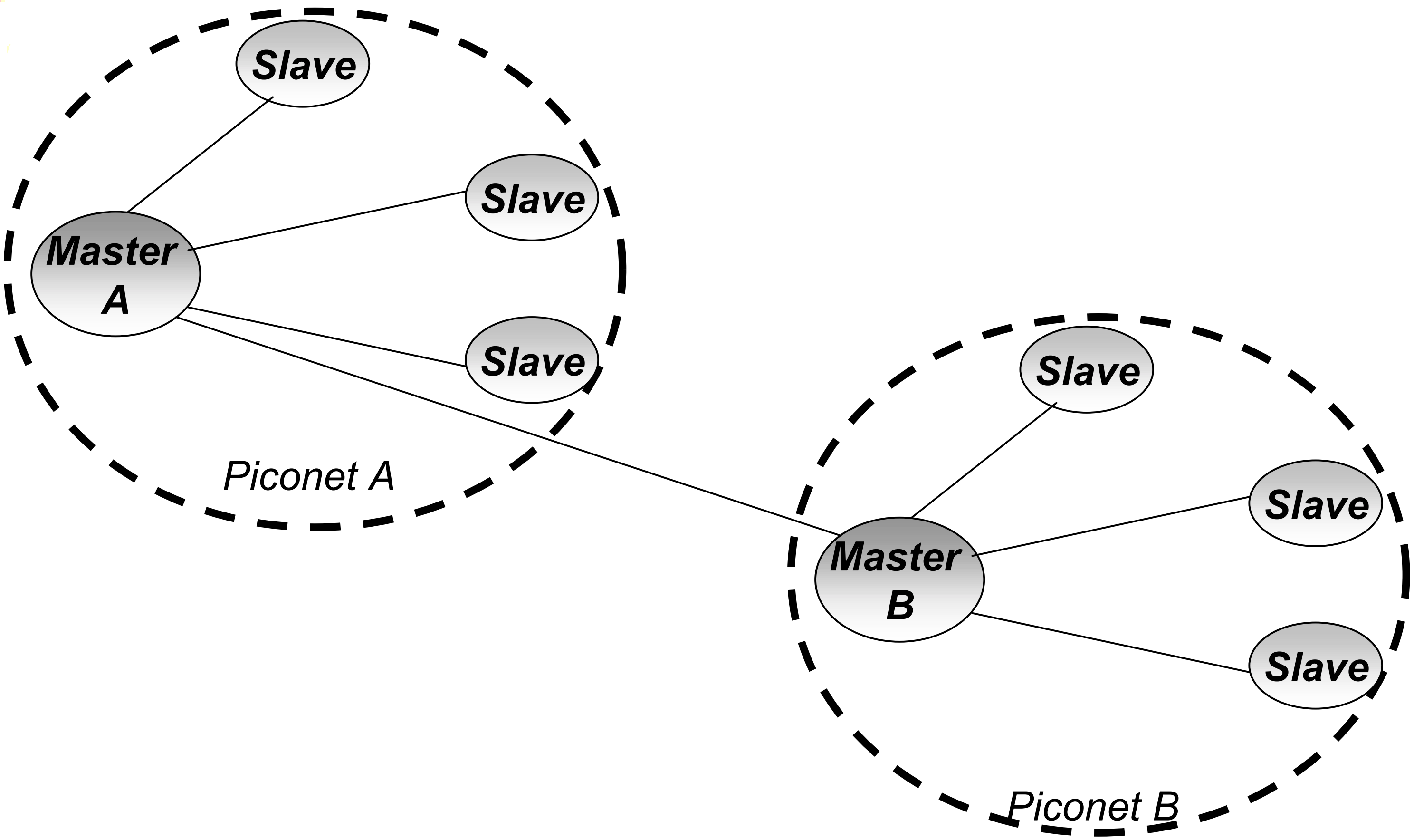
Wireless Flying Sensor Platform

- In the lab and then at the videoconference.

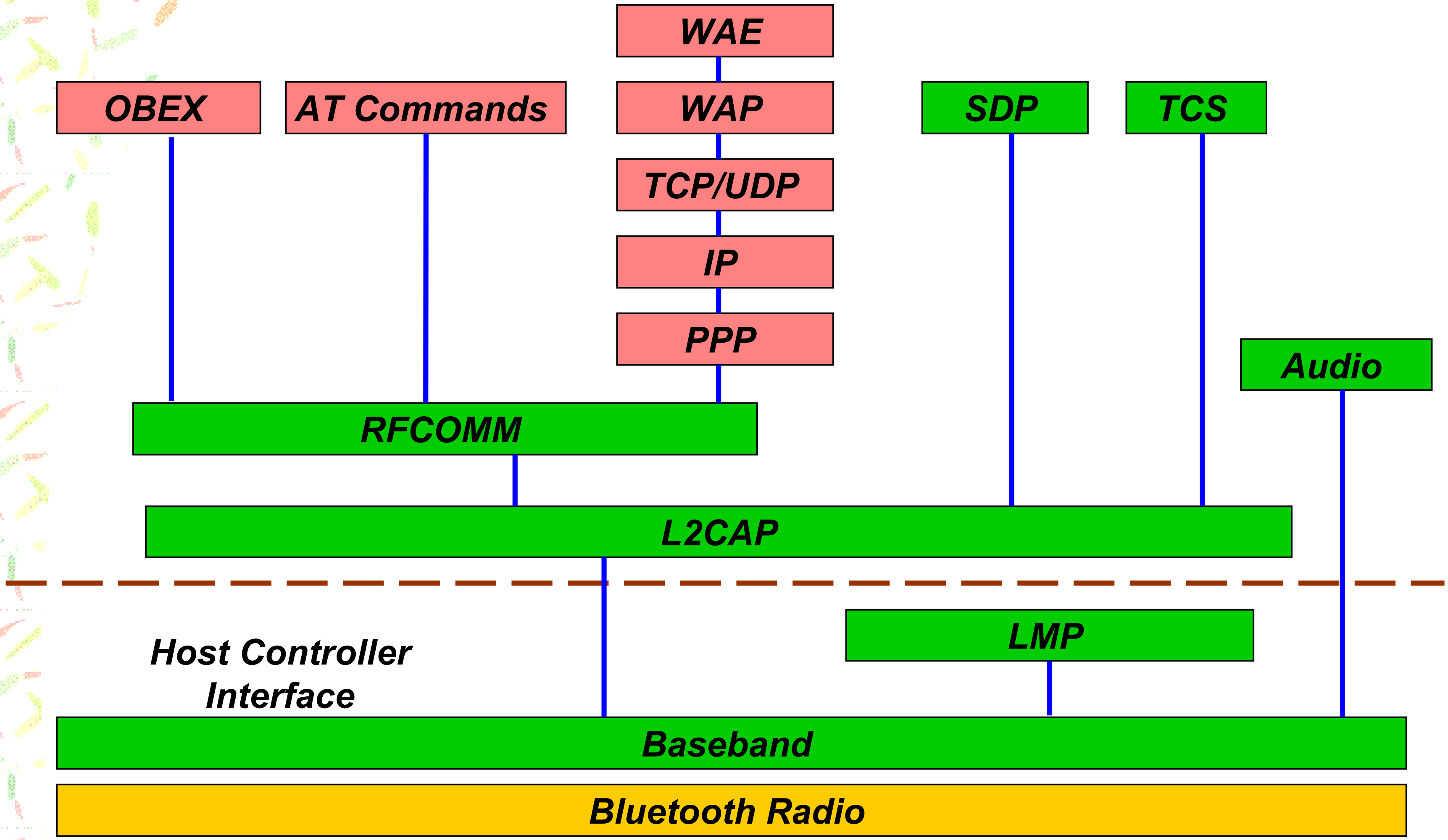
User Interface



Bluetooth Network Architecture



Bluetooth Protocol Stack



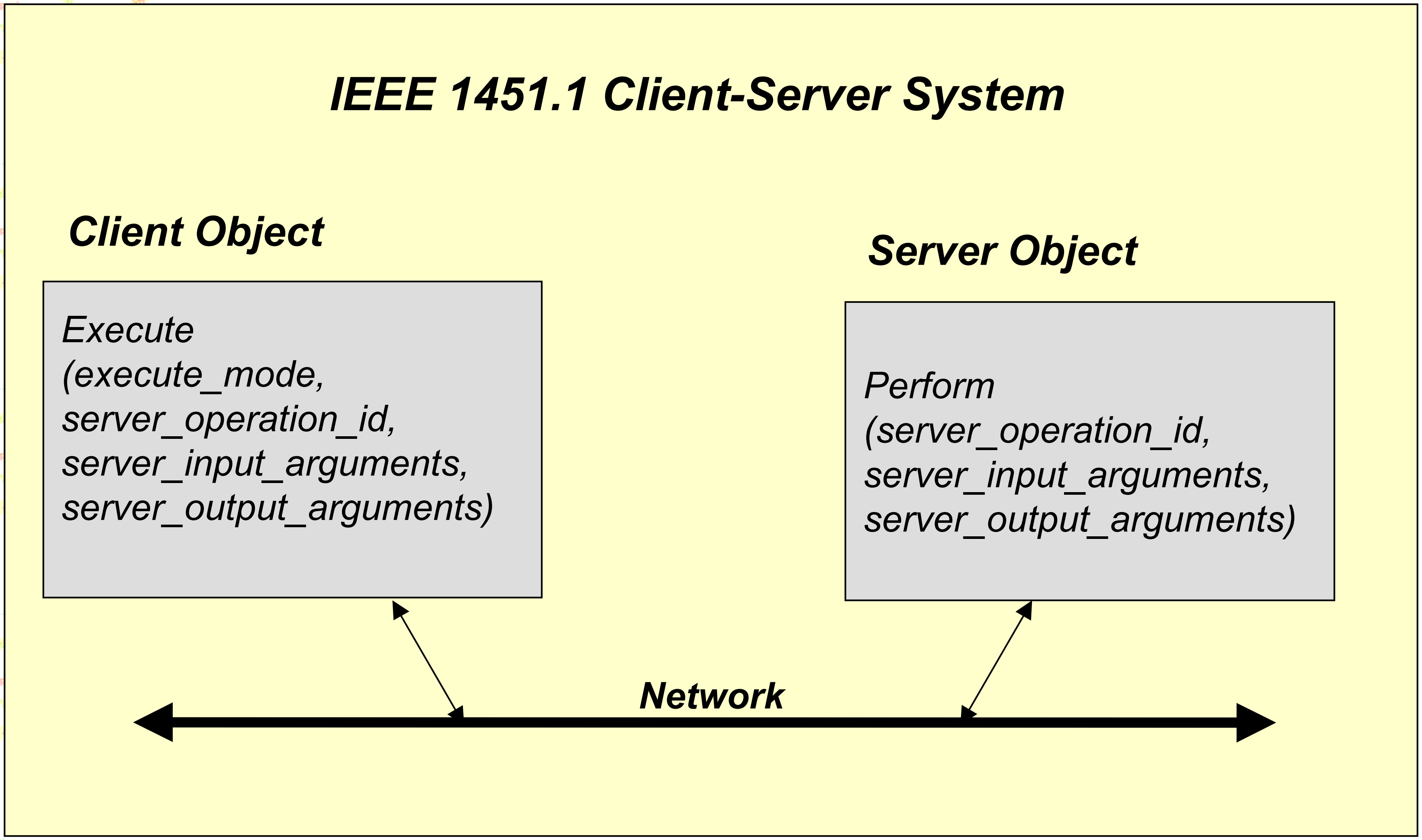
Protocol for interfacing Bluetooth with IEEE 1451

- IEEE 1451 Client nodes "Execute" the "Perform" operation on Server nodes
- Session-level protocol required for client server communication
- Two options
  - TCP/IP – Large overheads
  - OBEX – Light version of HTTP
- OBEX is the chosen alternative

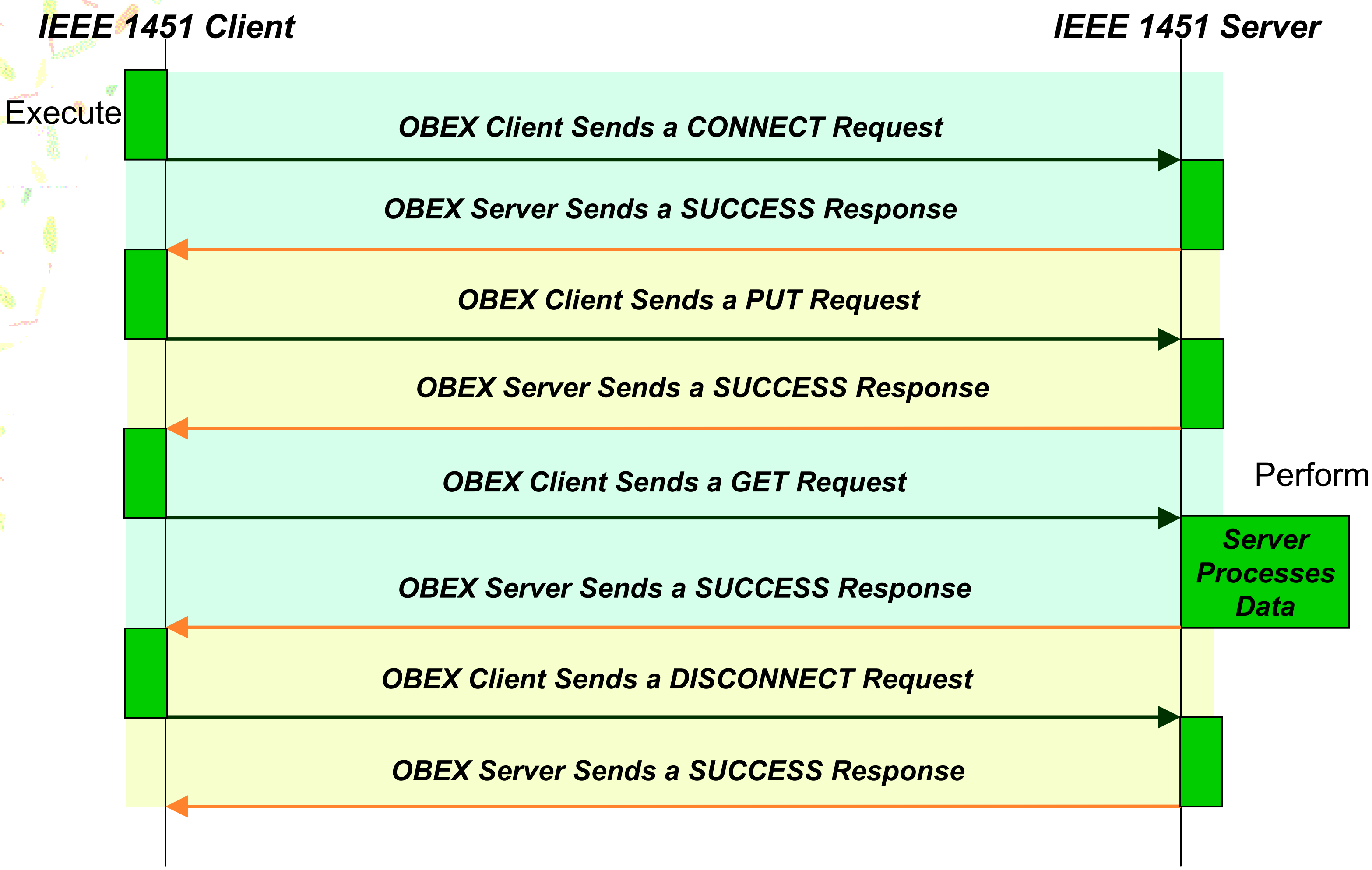
OBEX Protocol

- Primarily developed by IrDA and adopted by Bluetooth for interoperability
- Example applications are for short-range business card exchange or synchronization
- Can operate on both RFCOMM as well as TCP/IP as transport
- Defined by
  - OBEX Object Model
  - OBEX Session Protocol

IEEE 1451 Communication on Bluetooth

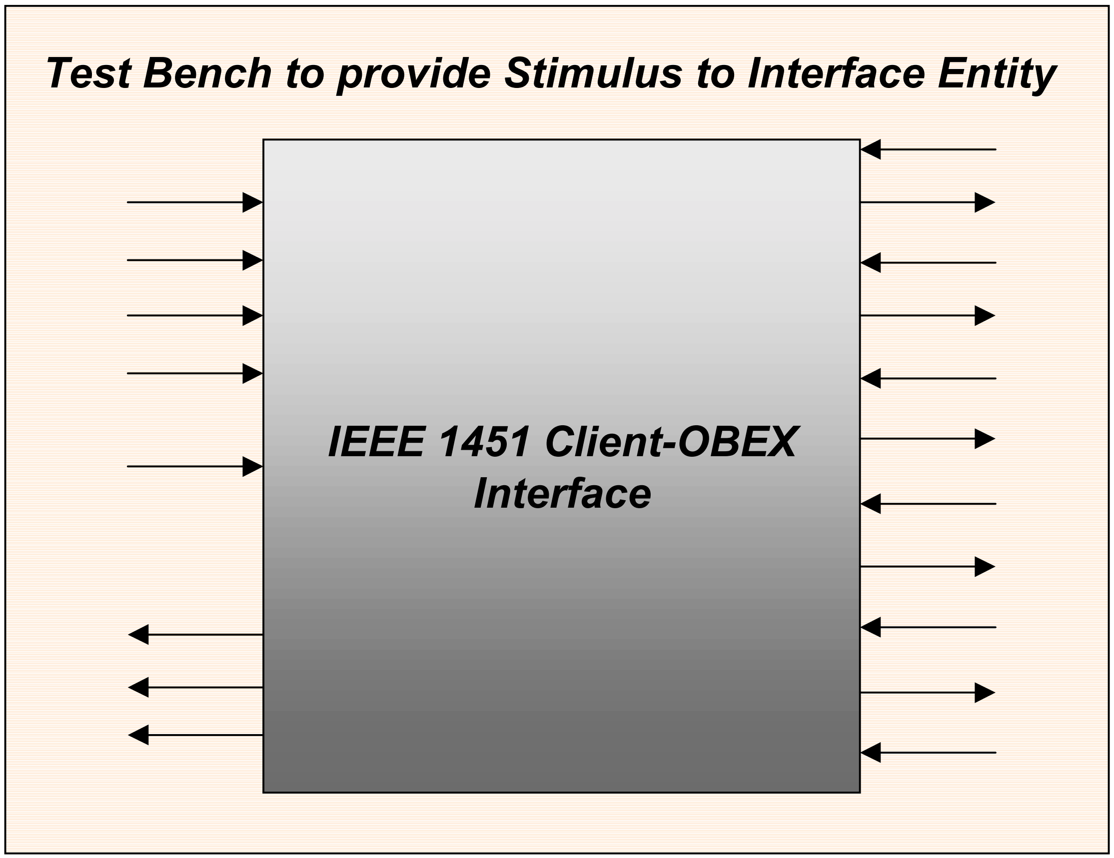


OBEX Session Protocol for Smart Transducers



Test Bench for Design

Simulates signals for providing Client requests and receiving Server response with error codes



Simulates signals for providing server responses in OBEX format

VHDL Design of Interface Entity

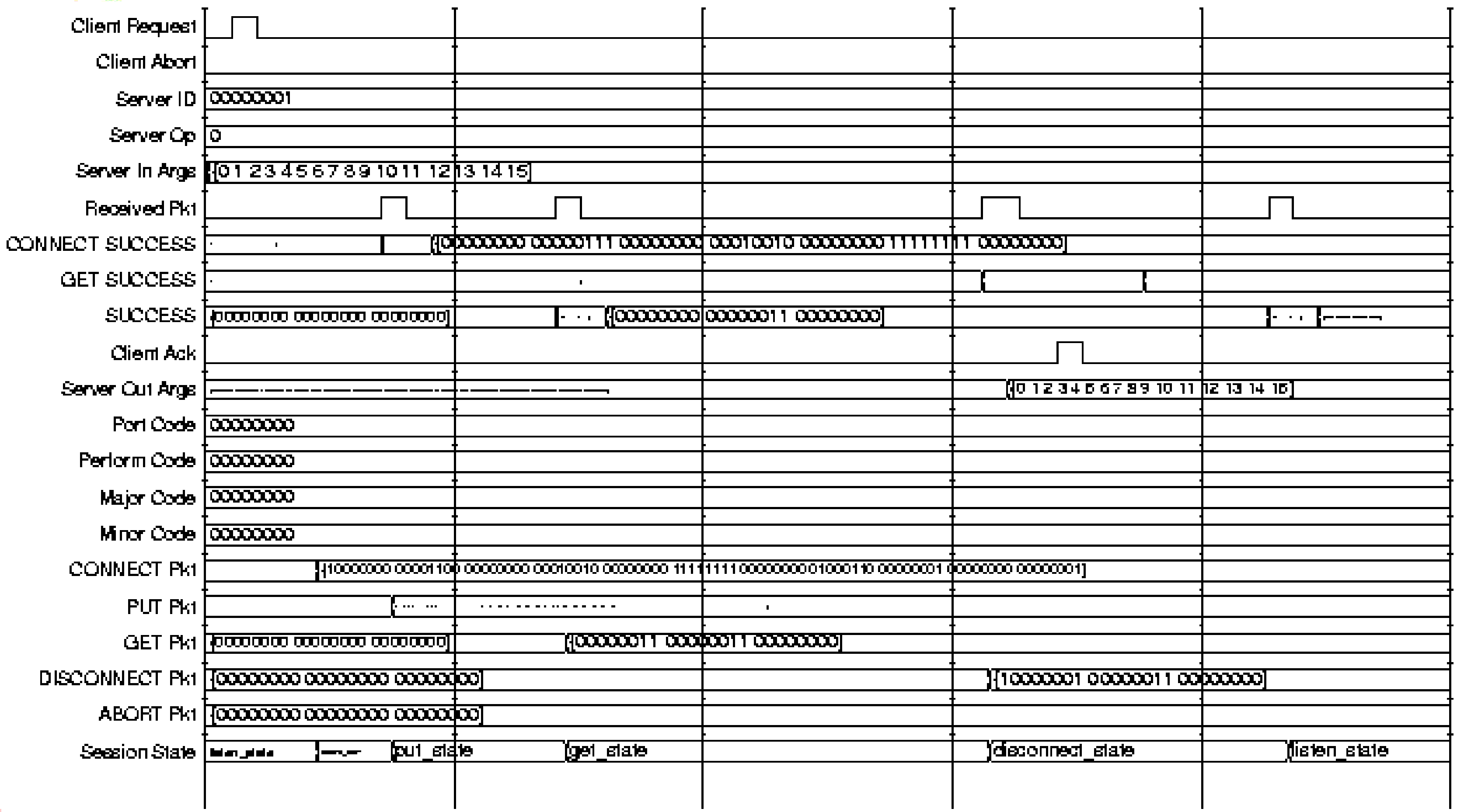
```
ENTITY interface IS
PORT(
   client_request : IN BIT;
   client_abort : IN BIT;
   serverid : IN UInteger8;
   server_operation : IN INTEGER;
   server_input_args : IN integer_array;
   received_pkt : IN BIT;
   connect_success_pkt : IN byte_array(0 TO 6);
   get_success_pkt : IN byte_array(0 TO payload_packet_length);
   success_pkt : IN byte_array(0 TO 2);
   client_ack : OUT BIT;
   server_output_args : OUT integer_array;
   portcode : OUT UInteger8;
   performcode : OUT UInteger8;
   majorcode : OUT UInteger8;
   minorcode : OUT UInteger8;
   connect_pkt : OUT byte_array(0 TO 10);
   put_pkt : OUT byte_array(0 TO payload_packet_length);
   get_pkt : OUT byte_array(0 TO 2);
   disconnect_pkt : OUT byte_array(0 TO 2);
   abort_pkt : OUT byte_array(0 TO 2);
   session_state : OUT state);
END interface;
```

VHDL Design of OBEX Session

```
ENTITY obex_session IS
  PORT(check : IN BIT; -- clock for session protocol
       client_abort : IN BIT; -- client node aborts operation
       server_id : IN UInteger8;
       obex_body_object : IN obex_body;
       obex_send : IN BIT;
       obex_receive : IN BIT;
       connect_success_pkt : IN byte_array(0 TO 6); -- success to connect
       get_success_pkt : IN byte_array(0 TO payload_packet_length); --for output
       success_pkt : IN byte_array(0 TO 2); -- success to put, disconnect, abort
       obex_result_object : OUT obex_result;
       session_state : OUT state; -- state of OBEX session communication
       connect_pkt : OUT byte_array(0 TO 10); -- start obex session with connect
       put_pkt : OUT byte_array(0 TO payload_packet_length); -- put obex body
       get_pkt : OUT byte_array(0 TO 2); -- get results from server
       disconnect_pkt : OUT byte_array(0 TO 2); -- end session
       abort_pkt : OUT byte_array(0 TO 2); -- stop operation in the middle of a session
       timeout_error : OUT BIT -- indicate session has timed out
       );
END obex_session;
```

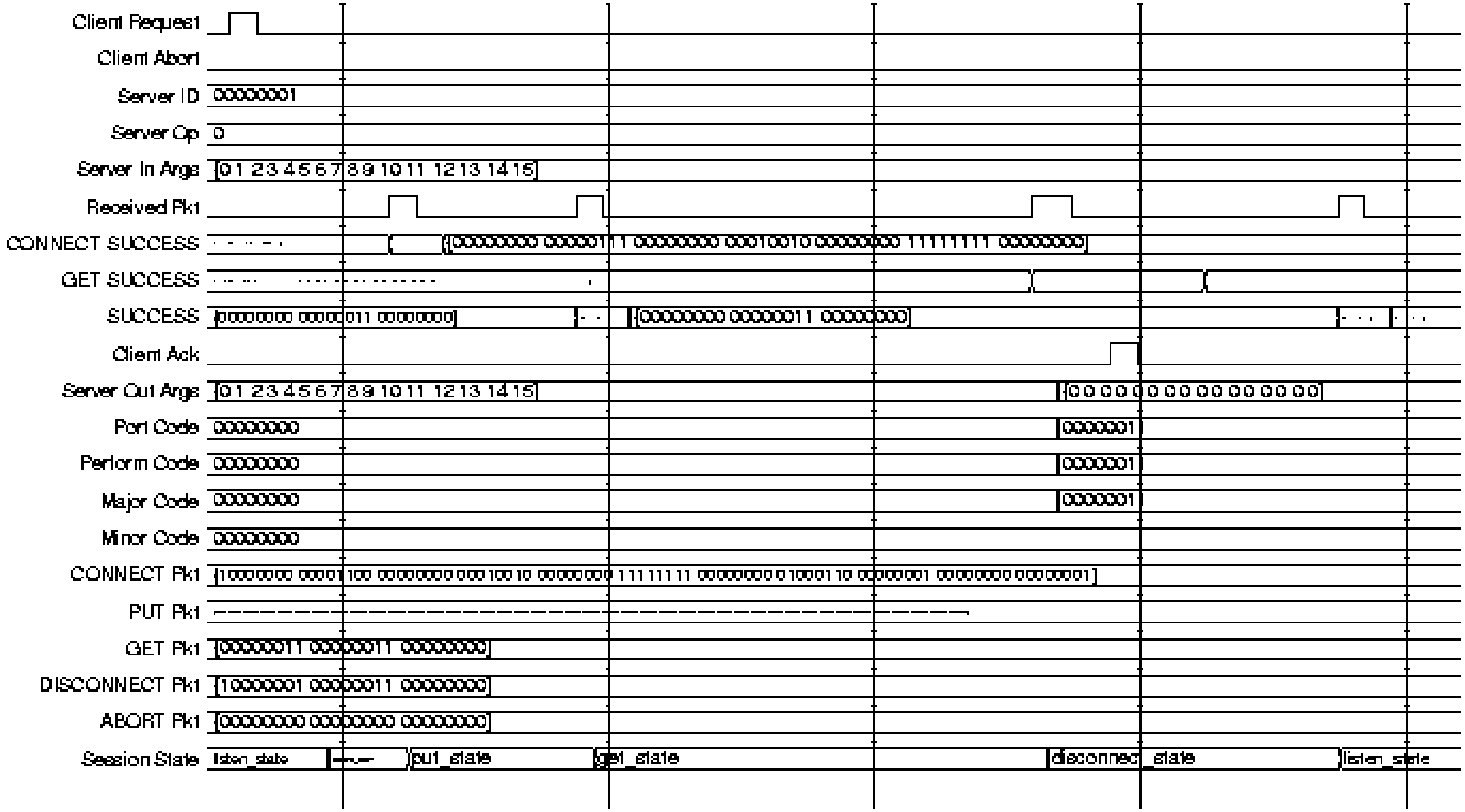
Design

Client-Server VHDL Simulation

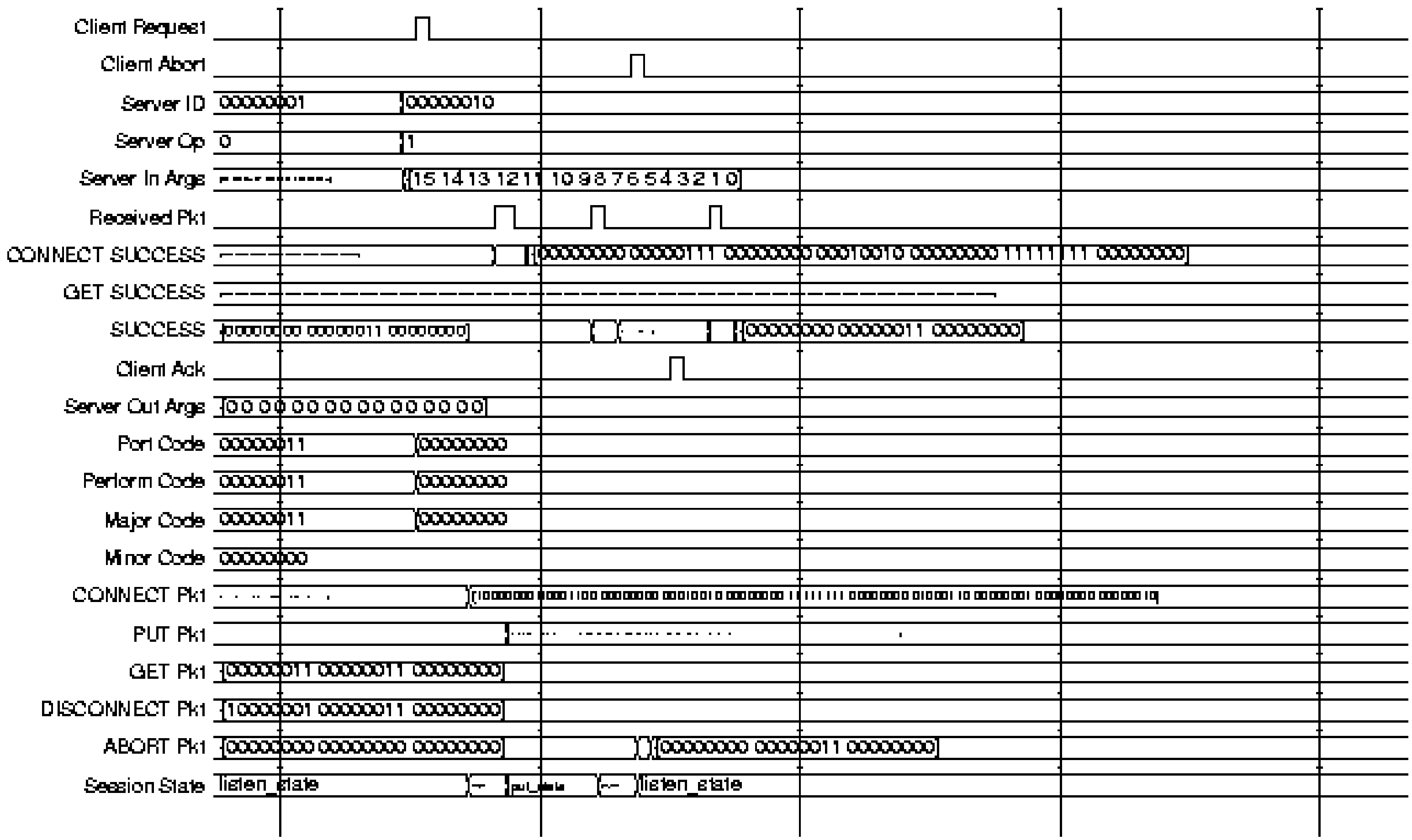


Design

Server Responds with error



Client Abort



Conclusions

- VHDL descriptions enables an implementation-oriented investigation of wireless sensor specifications.
- Easier to assess the feasibility and "cost" of various features by "what if" test bench scenarios.
- Easier to "visualize" complex behavior and catch potential problems.
- Prototyping for iterative specifications.